Particle Simulation in a Multiprocessor Environment

Jeffrey D. McDonald[1]

Report RNR-91-003, January 1991

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

Particle Simulation in a Multiprocessor Environment

Jeffrey D. McDonald[1]

Report RNR-91-003, January 1991

NAS Systems Division
Applied Research Branch
NASA Ames Research Center, Mail Stop T045-1
Moffett Field, CA 94035

January 4, 1991

Abstract. This report summarizes the work carried out at Eloret Institute in 1990 investigating the implementation of a particle simulation method on the Intel iPSC/860.

[1]The author is with Eloret Institute, 3788 Fabian Way, Palo Alto, California 94303. This work has been partially supported by a grant from the NAS Applied Research Branch

INTRODUCTION

Particle simulation methods continue to attract considerable attention, especially as a technique for analyzing low density, hypersonic re-entry flows. It is well known that these methods are expensive computationally and require large amounts of processor memory for simulations of practical interest. Recent studies have focused on reducing the computational cost of particle simulation methods through reformulation of underlying algorithms to permit vectorization[1,2]. The pursuit of substantial vectorization has been primarily motivated by the availability of current state of the art supercomputers whose performance capabilities rely principally on vector concepts. Examples of such machines are the Cray-2 and the Cray-Y/MP from Cray Research. As single processor performance begins to asymptote, attention is focusing on parallel processing as a means of providing increasingly powerful machines. Even the aforementioned Cray computers have multiple CPU's (4-8) although the primary source of performance continues to be through vectorization (factor of 10-20).

Powerful new machines are now available in the commercial market that promise large scale parallelism along with lower cost to performance ratios as compared to current supercomputers. Examples of such machines include the Thinking Machines Connection Machine CM-2 and the Intel iPSC-2/860. Particle simulation methods have already been implemented on these and other parallel machines[3,4,5]. These initial results are very promising, indicating that these parallel machines can provide an excellent platform for particle simulation computations.

Investigations during this contract period have examined the implementation of a general particle simulation code in both vector and multiprocessor environments. Focusing on a complete simulation code allows not only a measure of raw compute performance, but also evaluation other elements such as input/output (IO) performance and overall usefulness of the computer-code combination in an engineering context. A primary goal of this continuing study is to produce a code that is easily ported across a wide range of parallel computer architectures. The Intel iPSC-2/860, Cray-2, and Cray-Y/MP computers are chosen as initial target architectures because of their performance, proven suitability to particle methods, and availability. This research will provide a framework for performing larger scale simulations than is currently possible as next generation supercomputers become available.

SIMULATION DESCRIPTION

A gas in a particle simulation method is statistically represented by a number of particles followed over a number of discrete time steps using the following sub-steps: 1) particles are moved through space, each with their individual velocity, 2) the state of particles which have crossed boundary surfaces are modified appropriately, 3) random candidate collision pairs are chosen for all of space ensuring both particles in a pair occupy the same cell (a small volume of the physical domain), 4) collisions are statistically chosen via a collision selection rule, 5) pairs of particles having appropriate

ouput as well as file ouput for use as input to other available plotting packages.

Work by Brian Haas, a Stanford Graduate student, has recently led to chemical models compatible with vectorization[7] and these have recently been implemented into the current code, referred to as PSim3. This code is capable of addressing three dimensional problems involving arbitrary geometries, however, a simply cubical cell network is retained from previous work[1] at the present time limiting spatial resolution in solving some problems.

Note that a primary difference of the adopted approach from that of the well known DSMC method[8] is the use of non-spatially ordered sampling of candidates for a given event. For example, candidate collision pairs are collected for all space before selection of actual collisions from these candidates is initiated and the size of this sample may be arbitrary as long as it is known at the time of selection from the candidates[1]. This eliminates the need for a sort operation of particles into cells as is required when applying the time counter[8] or the no time counter (NTC)[9] algorithms of Bird. There the sample of candidate pairs is constrained to a calculated size in each cell and this sample size must be generated exactly. The arbitrary sample size approach was adopted primarily to allow for efficient vectorization[2] and is retained in the current work for the same reason. Vectorization and parallel processing are complementary and the objective here is to develop a truly portable code having a high performance on platforms with multiple processors, vector hardware, or ideally both.

TARGETED COMPUTER ARCHITECTURES

There is an ever increasing variety of multiple processor based computers being made available. One helpful classification makes a distinction between Single Instruction Multiple Data (SIMD) machines and Multiple Instruction Multiple Data (MIMD) machines. Another important classification dealing with multiple processor computers deals with memory system architecture. Shared memory systems allow all processors access to a single bank of global memory. Distributed memory systems associate a bank of memory with each processor and communication between processor is only possible via a interconnect network. Of course, machines may combine elements of shared and distributed memory in hybrid memory organizations.

It is not feasible with current software development technology to provide a singe particle simulation code that runs efficiently across all possible parallel machines architectures. Many unique programming techniques are required to effectively utilize many of the variety of available parallel computers.

Multiple instruction streams offer the greatest flexibility for the development of a particle simulation method because arbitrarily large effective vector lengths become difficult to fully utilize as complexity is added to the physical models employed. This is best understood by example. Out of a large collection of particles moved during a time step, a much smaller number may collide with other particles. Typically even a smaller number will interact with boundaries and yet a smaller number will chemically

react. Very infrequent events may underutilize a SIMD machine having a large effective vector length. For this reason the code is written assuming a MIMD architecture, however, vectorizable algorithms are retained to take advantage of any traditional vector hardware having modest vector lengths. This is important as to maintain a high performance capability on existing Cray supercomputers. No assumptions are made regarding the memory organization; only that processors may communicate with one another in an asynchronous fashion. Details of this communication vary from computer to computer and represents the sole machine dependent code module in PSim3.

PROGRAMMING MODEL

Since no assumptions are made about memory organization, communication between the processors must not rely on shared memory. As the Intel iPSC-2/860 is one of the first target machines and because it relies on message passing primitives for communication, an abstracted message passing model was chosen for all interprocessor communication. Message passing primatives are not inherent on Cray machines runing current versions of Unicos so these must be implemented using pipes, sockets, or shared memory.

The PSim3 code is written such that there is a host process running for the duration of the simulation. This host process handles the user interface and synchronizes activity of a number of node processes which perform the actual simulation. The host process communicates with nodes

via the message passing model. In the case of the iPSC-2/860, the host process runs on an actual front end computer based on an i386 processor while each node process has a dedicated i860 microprocessor and 8Mbytes of local memory. On the Cray machines the host process simply runs on any available processor as does each of the node processes. In the case of the Cray implementation the number of node process may exceed the number of available physical processors. The Cray implementation is equally capable of running on any standard Unix system having any number of processors that can run Unix processes. Note that the processing nodes on the iPSC-2/860 do not run Unix processes, rather each node runs a small message passing kernel. This requires special programming different from the Cray implementation, however, all machine dependent code is encapsulated in a single module as mentioned earlier.

PARALLEL DECOMPOSITION

The most important issue is how to divide the simulation computation across nodes processes using the single consistent message passing programming model defined above. Since communication can easily become a bottleneck in parallel applications when interconnect network and processor speeds are not balanced, it is desirable to exploit some form of data locality. The most strait forward source of such locality is to divide the spatial domain of the simulation into a number of sub-domains or regions equal to the desired number of node processes. Communication between processes occurs as a particle passes from one region to another. Particles

crossing region "seams" are treated simply as an additional type of boundary condition.

Replication of data structures across processes must be minimized as memory is a limited resource. The data representation of physical space, for example, must be distributed along with the particles that occupy that space. This is accomplished by surrounding each simulated region of space by a shell of extra cells that, when entered by a particle, directs that particle to the neighboring region. Nowhere can there exist a single representation of all simulated space as this would quickly become a serious bottleneck for large simulations performed using many regions. Load balancing is obtained by allowing region sizes and locations to change with time, however this has not yet been implemented.

PERFORMANCE RESULTS

Figure 1 presents initial results from the implementation of PSim3 including five specie air chemistry on the Intel iPSC-2/860. Performance is given by elapsed time in seconds per time step per simulated particle. Similar runs without chemistry or multiple processor support carried out on a single Cray-2 processor and a single Cray-Y/MP processor are presented. Testing has shown that chemistry models adds a 10% performance penalty when running the iPSC-2/860 code. Figure 2 presents the speedup obtained with the iPSC code as the number of processors utilized is doubles. It should be noted that the problem size is being doubled along with the number of processors and a uniform gas is being simulated at the present

time to avoid the load balancing issue. It is clear from these early results that the full Intel iPSC-2/860 machine would provide at least the performance of a single Cray-Y/MP processor and better than 2 Cray-2 processors. Note the speedup with 64 processors is 45 and the speedup curve is near linear. This indicates that many more processors could be dedicated to a particle simulation in an efficient manner.

CONCLUDING REMARKS

A very high performance, vectorized, general purpse particle simulation code has been extented to include the effect of multi-component gases. Additionally, vector compatible finite rate chemistry modeling has been included providing a very powerful and flexible simulation tool. Extensive graphics oriented support code has also been made available.

It also appears that particle simulation methods benefit greatly from the use of parallel machines. The new simulation code has been written in such a way as to be easy to port to new high performance MIMD machines as they become available. The code will very soon provide a good comparison of the cost/performance ratios of the Intel iPSC-2/860, the Cray-2, and the Cray-Y/MP, when applied to a particle simulation problem.

REFERENCES

[1] McDonald, J.D., "A Computationally Efficient Particle Simulation Method Suited to Vector Computer Architectures," Ph.D. Thesis, Department of Aeronautics and Astronautics, Stanford University, December 1989

[2] Baganoff, D. and McDonald, J.D.,"A Collision-Selection Rule for a Particle Simulation Method Suited to Vector Computers," Submitted to *Physics of Fluids* (1990)

[3] Dagum, Leonardo, "On the Suitability of the Connection Machine for Direct Particle Simulation" Ph.D. Thesis, Department of Aeronautics and Astronautics, Stanford University, June 1990

[4] Wilmoth, Richard G., "Direct Simulation Monte Carlo Analysis on Parallel Processors", AIAA Paper No. 89-1666 (1989)

[5] Furlani, T.R. and Lordi, J.A., "A Comparison of Parallel Algorithms for the Direct Simulation Monte Carlo Method-II: Application to Exhaust Plume Flowfields", AIAA Paper No. 89-1167 (1989)

[6] Feiereisen, W., McDonald, J.D., Fallavollita, M., "Three Dimensional Discrete Particle Simulation About The AFE Geometry", AIAA Paper No. 90-1778 (1990)

[7] Haas, B. L., "Fundamentals Of Chemistry Modeling Applicable To A Vectorized Particle Simulation," AIAA Paper No. 90-1749 (1990)

[8] Bird, G.A, *Molecular Gas Dynamics*, Claredon Press, (1976)

[9] Bird, G.A, "Perceptions of Numerical Methods in Rarefied Gas Dynamics", in *Rarefied Gas Dynamics*, edited by E.P. Muntz, D.P. Weaver,

and D.H. Campbell, Volume 118 of Progress in Aeronautics and Astronautics, AIAA, Washington, pp. 211-226, (1989).

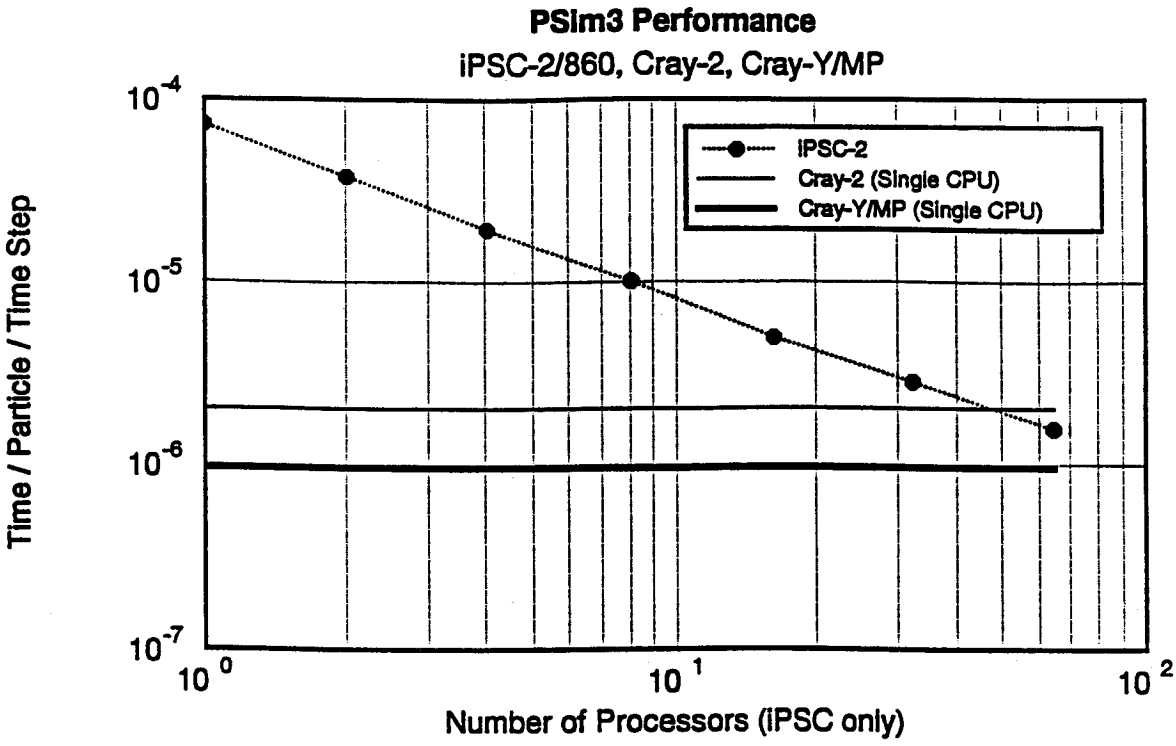

Figure 1) Initial PSim3 performance with chemistry on the Intel iPSC-2/860. Elapsed time in seconds per particle per time step. Problem size varies with number of processors. Cray-2 and Cray-Y/MP single processor times provided for comparison but using CPU time without chemistry.

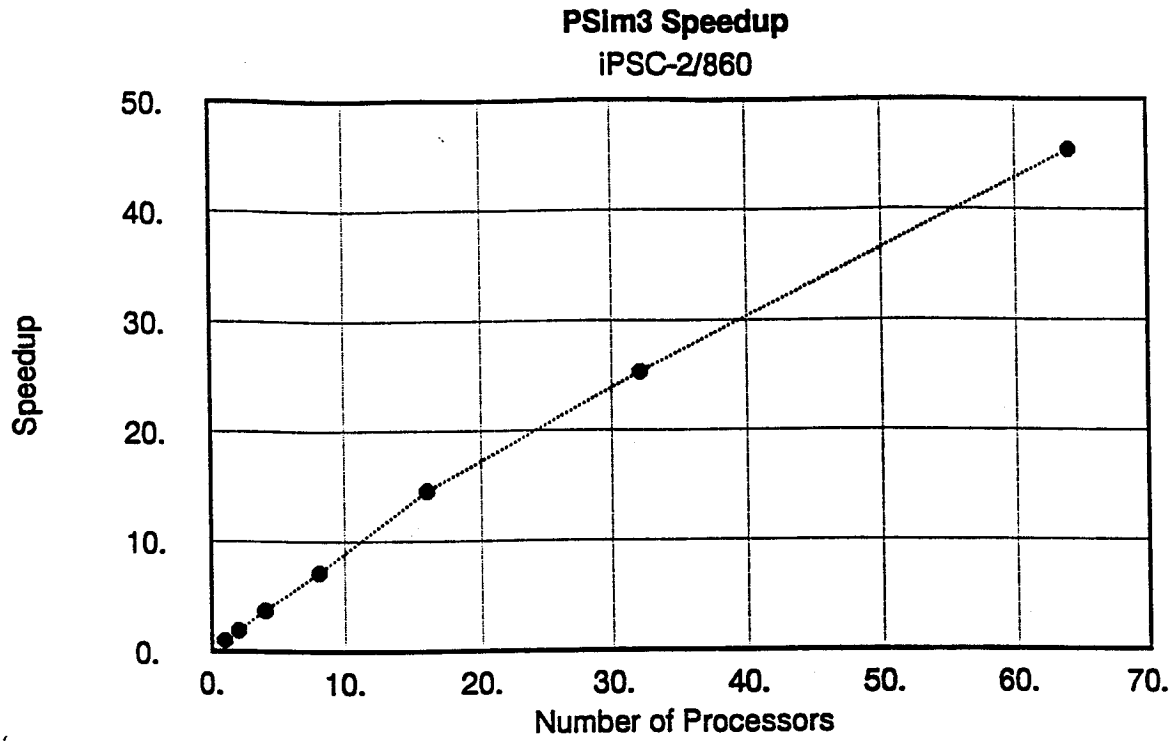

Figure 2) Initial results for speedup plotted against number of processors for PSim3 running on the Intel iPSC-2/860.

ABSTRACT SUBMITTAL

PARTICLE SIMULATION IN A MULTIPROCESSOR ENVIRONMENT

Jeffrey D. McDonald†
Eloret Institute, 3788 Fabian Way,
Palo Alto, California 94303

INTRODUCTION

Particle simulation methods continue to attract considerable attention, especially as a technique for analyzing low density, hypersonic re-entry flows. It is well known that these methods are expensive computationally and require large amounts of processor memory for simulations of practical interest[1]. Recent studies have focused on reducing the computational cost of particle simulation methods through reformulation of underlying algorithms to permit vectorization[2] The pursuit of substantial vectorization has been primarily motivated by the availability of current state of the art supercomputers that whose performance capabilities rely principally on vector concepts. Examples of such machines are the Cray-2 and the Cray-Y/MP from Cray Research. As single processor performance begins to asymptote, attention is focusing on parallel processing as a means of providing increasingly powerful machines. Even the aforementioned Cray computers have multiple CPU's (4-8) although the primary source of performance continues to be through vectorization (factor of 10-20).

Powerful new machines are now available in the commercial market that promise large scale parallelism along with lower cost to performance ratios as compared to current supercomputers. Examples of such machines include the Thinking Machines Connection Machine CM-2 and the Intel iPSC-2/860. Particle simulation methods have already been implemented on these and other parallel machines[4,5,6]. These initial results are very promising, indicating that these parallel machines can provide an excellent platform for particle simulation computations.

This paper to examines the implementation of a general particle simulation code in a multiprocessor environment. Focusing on a complete simulation code allows not only a measure of raw compute performance, but also evaluation other elements such as input/output (IO) performance and overall usefulness of the computer-code combination in an engineering context. A primary goal of this study is to produce a code that is easily ported across a wide range of parallel computer architectures. The Intel iPSC-2/860, Cray-2, and Cray-Y/MP computers are chosen as initial target architectures because of their performance, proven suitability to particle methods, and availability. It is hoped that this research will provide a framework for performing larger scale simulations than currently possible on next generation supercomputers.

SIMULATION DESCRIPTION

Recently, investigations into particle simulation techniques have been carried out by a group consisting of researchers from both Stanford University and NASA Ames research center. Until very recently, practical large-scale implementations of the adopted methods have been limited to non-reacting multicomponent gases. Emphasis had been on the restructuring of basic algorithms to enhance computational performance on vector computer architectures. Work by Haas has recently led to chemical models compatible with vectorization[7] and these have been implemented into the current code, referred to as PSim3. This code is capable of addressing three dimensional problems involving arbitrary geometries however a simply cubical cell network is retained from previous work[2] at the present time limiting spatial resolution in solving some problems.

A gas in a particle simulation method is statistically represented by a number of particles followed over a number of discrete time steps using the following sub-steps: 1) particles are moved through space, each with their individual velocity, 2) the state of particles which have crossed boundary surfaces are modified appropriately[2], 3) random candidate collision pairs are chosen for all of space ensuring both particles in a pair occupy the same cell[2], 4) collisions are statistically chosen via a collision selection rule[2,3], 5) pairs of particles having appropriate type for a user specified reaction are considered for that reaction, and reacted on a statistical basis[7], 6) collision pairs that have not reacted undergo thermal collisions[2], 7) samples are optionally taken to form macroscopic results. Details concerning each of these sub-steps are beyond the scope of this paper and are covered in the cited references.

Note that a primary difference of this approach from that of the well known DSMC method[8] is the use

† Research Scientist, Member, AIAA. Mailing Address: NASA Ames Research Center, MS 230-2, CA 94035.

of non-spatially ordered sampling of candidates for a given event. For example, candidate collision pairs are collected for all space before selection of actual collisions from these candidates is initiated and the size of this sample may be arbitrary as long as it is known at the time of selection from the candidates[2]. This eliminates the need for a sort operation of particles into cells as is required when applying the time counter[8] or the no time counter (NTC)[9] algorithms of Bird. There the sample of candidate pairs is constrained to a calculated size in each cell and this sample size must be generated exactly. The arbitrary sample size approach was adopted primarily to allow for efficient vectorization[3] and is retained in the current work for the same reason. Vectorization and parallel processing are complementary and the objective here is to develop a truly portable code having a high performance on platforms with multiple processors, vector hardware, or ideally both.

TARGET ARCHITECTURES

There is an ever increasing variety of multiple processor based computers being made available. One helpful classification makes a distinction between Single Instruction Multiple Data (SIMD) machines and Multiple Instruction Multiple Data (MIMD) machines. Each of these is now briefly examined

The first, SIMD, describes the case where a single issued machine instruction is intended to operate simultaneously in the same way on a number of distinct data elements. The larger the number of data elements the larger the degree of parallelism. Vector processors are often classified as SIMD although there the single instruction does not act simultaneously across data elements, rather a "vector" of elements are operated on in a highly efficient pipelined fashion. High end vector machines typically also provide high scaler performance which reduces the impact of small serial code segments that cannot be vectorized. Massively parallel SIMD architectures such as the Connection Machine are similar to vector processors except the effective vector length is very long (up to 64K for the CM-2) and operations are actually simultaneous as there is a separate processor for each data element. Because the effective vector length is long and the individual processors are typically not very powerful, it is very important to minimize serial code segments or code segments using only a small fraction of the available vector size on SIMD computers such as the Connection Machine.

The second class, MIMD, describes the case where a number of processors have the ability to communicate with one another but are free to execute distinct instruction streams. There are a wide variety of machines in this category that span a broad performance range, from personal computer add-in boards and workstations to supercomputers.

Another important classification dealing with multiple processor computers deals with memory system architecture. Shared memory systems allow all processors access to a single bank of global memory. This situation is depicted in figure 1(a) where inter-processor communication may be via shared memory and/or a separate inter-processor communications network. Distributed memory systems associate a bank of memory with each processor and communication between processor is only possible via a interconnect network as shown in figure 1(b). Of course, machines may combine elements of shared and distributed memory as illustrated in figure 1(c).

Even the very brief introduction given above, concerning parallel machine classification, makes it very clear that it is not feasible with current software development technology to provide a singe particle simulation code that runs efficiently across all possible parallel machines. Many unique programming techniques are required to effectively utilize many of the variety of available parallel computers.

Multiple instruction streams offer the greatest flexibility for the development of a particle simulation method because arbitrarily large effective vector lengths become difficult to fully utilize as complexity is added to the physical models employed. This is best understood by example. Out of a large collection of particles moved during a time step, a much smaller number may collide with other particles. Typically even a smaller number will interact with boundaries and yet a smaller number will chemically react. Very infrequent events may underutilize a SIMD machine having a large effective vector length. For this reason the code is written assuming a MIMD architecture however vectorizable algorithms are retained to take advantage of any traditional vector hardware having modest vector lengths. This is important as to maintain a high performance capability on existing Cray supercomputers. No assumptions are made regarding the memory organization; only that processors may communicate with one another in an asynchronous fashion. Details of this communication will vary from computer to computer and represents the sole machine dependent code module in PSim3.

PROGRAMMING MODEL

Since no assumptions are made about memory organization, communication between the processors must not rely on shared memory. As the Intel iPSC-2/860 is one of the first target machines and because it relies on message passing primitives for communication, an abstracted message passing model was chosen for all interprocessor communication. Message passing primatives are not inherent on Cray machines runing current versions of Unicos so these must be implemented using pipes, sockets, or shared memory. Sockets have initially been chosen because they offer more flexibility

than the other primatives .

The PSim3 code is written such that there is a host process running during the simulation. This host process handles the user interface and synchronizes activity of a number of node processes which perform the actual simulation. The host process also communicates with nodes via the message passing model. In the case of the iPSC-2/860, the host process runs on an actual front end computer based on an i386 processor while each node process has a dedicated i860 microprocessor and 8Mbytes of memory. On the Cray machines the host process simply runs on any available processor as does each of the node processes. In the case of the Cray implementation the number of node process may exceed the number of available physical processors. The Cray implementation is equally capable of running on any standard Unix system having any number of processors that can run Unix processes. Note that the processing nodes on the iPSC-2/860 do not run Unix processors rather each node runs a small message passing kernel. This requires special programming different from the Cray implementation, however, all machine dependent code is encapsulated in a single module as mentioned earlier.

PARALLEL DECOMPOSITION

The most important issue is how to divide the simulation computation across nodes processes using the single consistent message passing programming model defined above. Since communication can easily become a bottleneck in parallel applications when interconnect network and processor speeds are not balanced, it is desirable to exploit some form of data locality. The most strait forward source of such locality is to divide the spatial domain of the simulation into a number of sub-domains or regions equal to the desired number of node processes. Communication between processes occurs as a particle passes from one region to another. This is the method used by Wilmoth[5] and it is also used in this work to minimize communication. Particles crossing region "seams" are treated simply as an additional type of boundary condition.

Replication of data structures across processes must be minimized as memory is a limited resource. The data representation of physical space, for example, must be distributed along with the particles that occupy that space. This is accomplished by surrounding each simulated region of space by a shell of extra cells that, when entered by a particle, directs that particle to the neighboring region. Nowhere can there exist a single representation of all simulated space as this would quickly become a serious bottleneck for large simulations performed using many processors. Figure 2 depicts a decomposition of a physical domain into four smaller regions, demonstrated in two dimensions for simplicity.

Load balancing is obtained by allowing region sizes and locations to change with time. There is a constant number of regions along each spatial dimension and each of their shapes corresponds to a rectangular parallelepiped. A region can be altered in size by only a single cell in each spatial dimension during a single adaption step. This greatly simplifies the adaption procedure when using a distributed representation of space. The computational and communication costs of such an adaption are expected to be low enough to allow it to be performed every $10-25$ steps during the transient phase of a simulation.

PERFORMANCE RESULTS

Figure 3 presents initial results from the implementation of PSim3 including five specie air chemistry on the Intel iPSC-2/860. Performance is given by elapsed time in seconds per time step per simulated particle. Similar runs without chemistry or multiple processor support carried out on a single Cray-2 processor and a single Cray-Y/MP processor are presented. Testing has shown that chemistry models adds a 10% performance penalty when running the iPSC-2/860 code. Figure 4 presents the speedup obtained with the iPSC code as the number of processors utilized is doubles. It should be noted that the problem size is being doubled along with the number of processors and a uniform gas is being simulated at the present time to avoid the load balancing issue. It is clear from these early results that the full Intel iPSC-2/860 machine would provide at least the performance of a single Cray-Y/MP processor and better than 2 Cray-2 processors.. Note the speedup with 64 processors is 45 and the speedup curve is near linear. This indicates that many more processors could be dedicated to a particle simulation in an efficient manner.

CONCLUDING REMARKS

From early results it appears that particle simulation methods benefit greatly from the use of parallel machines. The new simulation code has been written in such a way as to be easy to port to new high performance MIMD machines as they become available. The code will very soon provide a good comparison of the cost/performance ratios of the Intel iPSC-2/860, the Cray-2, and the Cray-Y/MP, when applied to a particle simulation problem.

REFERENCES

[1] Feiereisen, W., McDonald, J.D., Fallavollita, M., "Three Dimensional Discrete Particle Simulation About The AFE Geometry", AIAA Paper No. 90-1778 (1990)

[2] McDonald, J.D., "A Computationally Efficient Particle Simulation Method Suited to Vector Computer Architectures," Ph.D. Thesis, Department of Aeronautics and Astronautics, Stanford University, December 1989

[3] Baganoff, D. and McDonald, J.D.,"A Collision-Selection Rule for a Particle Simulation Method Suited to Vector Computers," Submitted to *Physics of Fluids* (1990)

[4] Dagum, Leonardo, "On the Suitability of the Connection Machine for Direct Particle Simulation" Ph.D. Thesis, Department of Aeronautics and Astronautics, Stanford University, June 1990

[5] Wilmoth, Richard G., "Direct Simulation Monte Carlo Analysis on Parallel Processors", AIAA Paper No. 89-1666 (1989)

[6] Furlani, T.R. and Lordi, J.A., "A Comparison of Parallel Algorithms for the Direct Simulation Monte Carlo Method II: Application to Exhaust Plume Flowfields", AIAA Paper No. 89-1167 (1989)

[7] Haas, B. L., "Fundamentals Of Chemistry Modeling Applicable To A Vectorized Particle Simulation," AIAA Paper No. 90-1749 (1990)

[8] Bird, G.A, *Molecular Gas Dynamics*, Claredon Press, (1976)

[9] Bird, G.A, "Perceptions of Numerical Methods in Rarefied Gas Dynamics", in *Rarefied Gas Dynamics*, edited by E.P. Muntz, D.P. Weaver, and D.H. Campbell, Volume 118 of Progress in Aeronautics and Astronautics, AIAA, Washington, pp. 211-226, (1989).

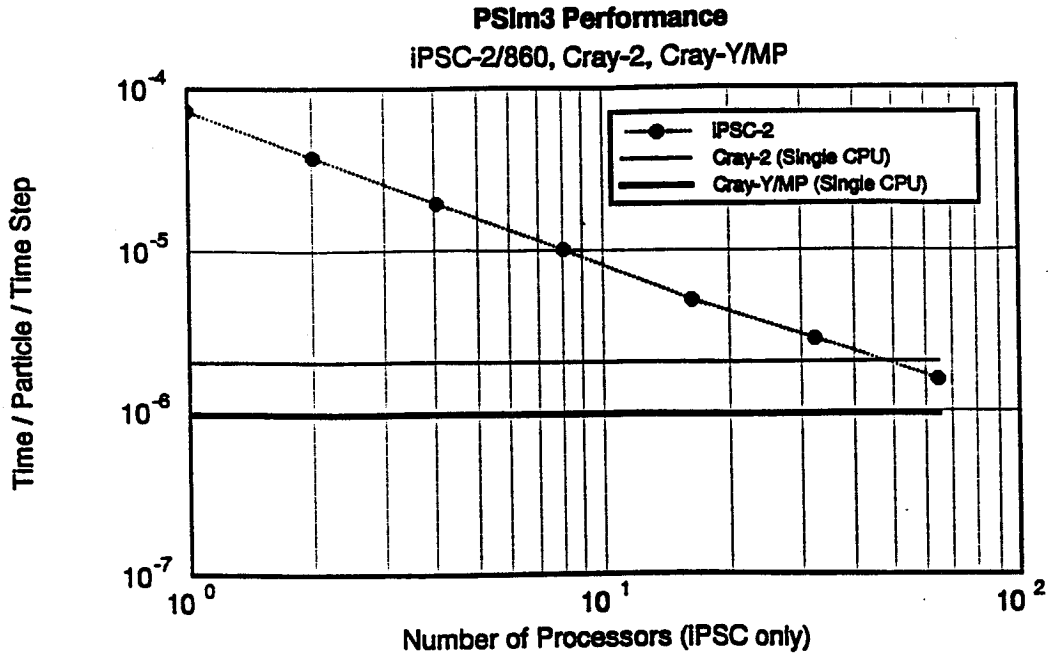

Figure 3) Initial PSim3 performance with chemistry on the Intel iPSC-2/860. Elapsed time in seconds per particle per time step. Problem size varies with number of processors. Cray-2 and Cray-Y/MP single processor times provided for comparison but using CPU time without chemistry.

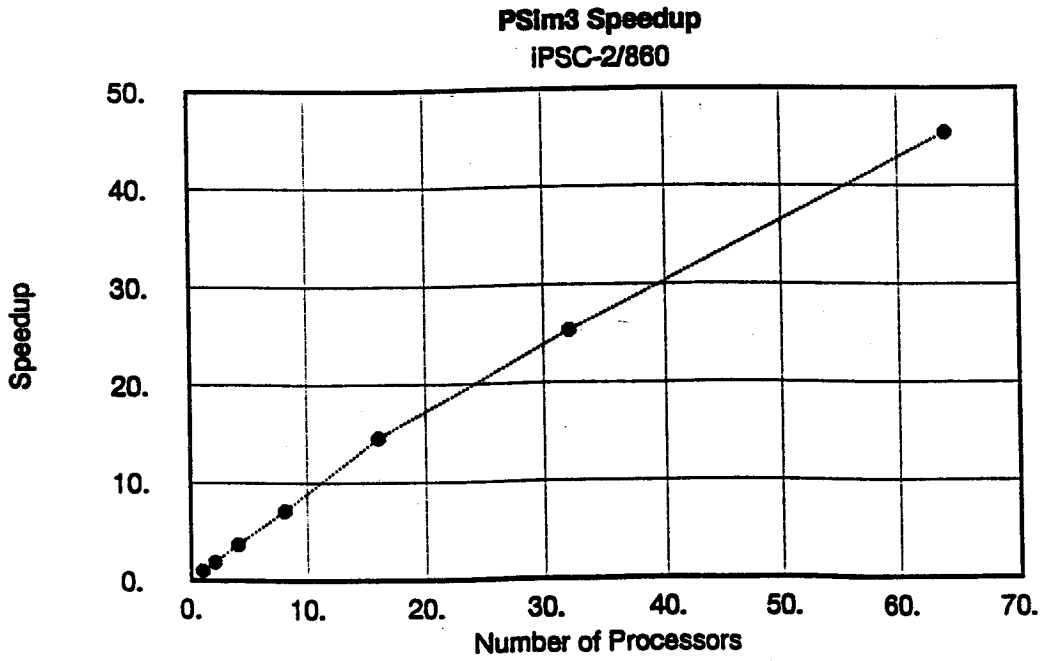

Figure 4) Initial results for speedup ploted against number of processors for PSim3 running on the Intel iPSC-2/860.

Three Dimensional Discrete Particle Simulation about the AFE Geometry

William J. Feiereisen*
Aerothermodynamics Branch, NASA Ames Research Center

Jeffrey D. McDonald**
Eloret Institute, Palo Alto, California

Michael A. Fallavollita***
Department Aeronautics and Astronautics, Stanford University

Abstract

The Discrete Particle Simulation method, due to Baganoff, has recently been extended to allow represention of gases composed of multiple species, to general power-law molecular interactions and to permit flows in thermal non-equilibrium. Particular attention has been paid to the implementation of this physics while retaining the efficiency of the original algorithm. Here, the enhanced algorithm is applied to the simulation of the flow field about the Aeroassisted Flight Experiment (AFE) vehicle with the same flight parameters as in a previous paper. The enhancements to the algorithm are introduced and comparisons are made to the previous calculation.

Introduction

Renewed interest in hypersonic flight in the upper atmosphere has sparked a renaissance of development of simulation methods for rarefied flows. Whereas continuum methods are applicable and efficient at the high densities associated with lower altitudes, it is appropriate to use particle simulation methods in the free molecular regime at very high altitude and in the so-called transition regime where the domains of applicability begin to overlap. Modern particle simulation methods have been developed by several investigators,[1,2,3] however the most widely used is the Direct Simulation Monte-Carlo (DSMC) method due to Bird.[4] This method has also been applied to the simulation of the flow field about the NASA Aeroassisted Flight Experiment (AFE)[5,6]. Unfortunately the structure of the algorithm in references 5 and 6 requires enormous computer resources limiting its frequent application. Computer time requirements have been the limiting factor in the use of this algorithm rather than the physical size of the problem as expressed by memory requirements.

In the last four years, work has progressed at Stanford University and NASA Ames Research Center[7,8,9,10] to reformulate the basic ideas inherent in the DSMC method in order to take advantage of modern vector and parallel computer architectures.

The goal has been to achieve at least an order of magnitude greater performance in order to handle a correspondingly larger number of particles than has been previously possible. This goal has been reached in previous papers with the intentional neglect of some of the physics known to be important in this flight regime. In particular, the gas model consisted of a single diatomic species. Although the simulated particles contained both translational and rotational energy, the algorithm assumed that these energy modes were close to equilibrium. The only relaxation model represented in the energy transfer to rotational energy was a non-physical computational peculiarity inherent in the model for the collision mechanics. There was previously no attempt made to model the vibrational energy modes. In large applications, intermolecular potentials were assumed to be "hard sphere" which is known to be a harder potential and result in less viscous behavior than real gases. The collision mechanics were performed in a unique and efficient way by the "Shuffle Algorithm"[7,11]. Unfortunately, this algorithm has proven difficult to extend to mixtures of gases and did not easily allow for modeling of vibrational modes. In order to ensure vectorization of some critical parts of the algorithm small portions of the original code were written in Cal-2 assembler limiting the portability of the code to the

* Research Scientist, Member AIAA
** Research Scientist, Member AIAA. Mailing Address: NASA Ames Research Center, MS-230-2, CA 94035
*** Student Member AIAA

Cray-2. The efficiency of the original algorithm, however, was quite good, allowing computation times less than 2 μsec per particle per time step on a Cray-2. Calculations with up to 10 million particles could be performed in less than five hours of CPU time.

In the last year much progress has been made in enhancing the physical modeling while retaining the basic efficiency of the algorithm. The calculation of the collision probability has been extended to general power law potentials allowing more realistic representation of viscous behavior. The previous collision algorithm derived much of its efficiency from the symmetry of representing all energy modes as velocities[7,11]. Rotational and vibrational energy modes for diatomic species are now represented directly instead of by using rotational velocity components. Now that this symmetry has been removed, additional problems associated with collisions between species of differing masses can be resolved by directly introducing isotropic scattering. Energy transfer to rotation and vibration are now represented by additional probabilities that model the relaxation to these quantities. Multiple species are now represented efficiently, and the implementation of chemical reactions amongst them, while retaining high performance, will be be the subject of another paper.

The improvements to the method will be outlined and then its application to the simulation of the flow field about the AFE flight vehicle will be described. Comparisons will be made to calculations made with the previous version of the method.

Method

Recent applications of the Stanford particle simulation method have been limited to single component monatomic or diatomic gases[10]. This permitted simplifications that allowed more rapid development of other essential aspects of the method, such as the collision selection algorithm. Recent extensions of the method include an ability to address multiple component gases exhibiting general power law molecular interactions as well as a more general treatment of rotational relaxation and the inclusion of vibrational energy modes. In the course of adding these elements to the simulation method, several changes to the algorithm have been made. These will be briefly reviewed here with reference to more detailed descriptions.

The collision selection algorithm used in the Stanford particle method has the advantage of being completely vectorizable[8]. The applicability of the basic algorithm has been extended to general gas mixtures while retaining vector computer architecture compatibility[11]. The resulting selection probability for collision of a particle of specie a and a particle of specie b from a sample size of S_{ab} candidate collision pairs having the specie combination ab is given by

$$P_s = \frac{n_a n_b}{(1+\delta_{ab})S_{ab}} \beta d_{ab}^2 g^{-\frac{1}{\alpha_{ab}}} g \Delta t$$

where n_a and n_b are specie number densities, βd_{ab}^2 is a simulation normalized reference collision cross section, g is the relative collision speed, α_{ab} is the assumed power law exponent, and Δt is the duration of the time step. Detailed descriptions of the analysis and implementation for this collision selection rule are available, as are a variety of techniques for assembling a collection of candidate collision pairs[8,11].

The algorithm to implement collision mechanics has changed radically since the previous AFE results were presented[10]. These changes were made necessary by two factors; namely the difficulty of extending the previous collision mechanics tocollisions between particles having different numbers of atoms and the inclusion of vibrational energy modes. Each of these issues are now briefly examined, however, reference 11 presents these in more detail.

The original shuffle algorithm[7,8,9,10] operated by creating a random signed permutation of the relative velocity vector components between the colliding particles. This generated a new post collision velocity state while insuring conservation of momentum and energy. This necessitated the representation of the rotational energy of a diatomic particle by a two component velocity vector. During a collision both internal and translational velocity states could be exchanged by the shuffle algorithm when the rotational modes were to relax. The collision of a monatomic particle with a diatomic particle presents a serious difficulty when using this algorithm. In such a collision, the complete velocity state of each particle has a different number of components and the definition of a relative velocity state is not obvious. The inclusion of vibrational energy presents similar problems. Because vibrational energy cannot typically be considered fully excited, a velocity representation analogous to rotation is not possible. Because of these issues, and the fact that modifications to the shuffle algorithm would prove computationally expensive, a more traditional

approach is now employed.

Isotropic scattering consistent with the variable hard sphere (VHS) model[13] is expensive due to the required evaluation of transcendental functions in selecting random post collision relative velocity vectors. Experience with the "Shuffle Algorithm" has shown that a discrete post collision scattering model is an adequate description in a particle simulation. This has been demonstrated even with extremely coarse discretizations of isotropic scattering. As such, it has proven adequate and very efficient to generate a large number of unit direction vectors chosen from an isotropic scattering distribution and choose from these at random for each collision[11]. The amount of memory used for such a table driven scheme is trivial compared to other memory intensive elements of the simulation.

Use of discretized isotropic scattering for translational modes requires a complementary technique for addressing internal energy modes. The Borgnakke-Larsen phenomenological model has been in widespread use for dividing energy between translational and internal energy modes since its inception[12]. This method has computationally expensive elements and it is desirable to consider the use of a discretized table driven variant of it for the sake of efficiency. This is not possible in general because of the strong temperature dependence of the vibrational energy modes. It is possible however to address the rotational modes with a table lookup scheme while separately managing the vibrational modes with a new model as described in detail in reference 11. Briefly, the post collision vibrational state is obtained through an iterative procedure which first utilizes rotational energies of the colliding particles as samples from a continuous two degree of freedom system. This may be quantized by the characteristic energy for vibration for the specie to arrive at a new vibrational state for the particle. After subtracting the sampled vibrational energy from the total collision energy, the remaining energy is then divided between translation and rotation using the Borgnakke-Larsen model assuming a fixed, two degree of freedom internal energy system for rotation. Since the number of internal modes is fixed for rotation, the distribution for the division of energy between relative translation and rotation may be tabulated. This sequence of steps may be repeated to remove correlation between rotational and vibrational energy modes. Three iterations has proven to be adequate even in highly non-equilibrium situations[11]. Once again, the amount of memory used for such a table driven scheme is trivial compared to other memory intensive elements of the simulation.

Since the algorithm was initially developed around a gas consisting of a single monatomic or diatomic species, there were simplifications that allowed construction of a very efficient code. For example, because all particles had the same mass by default, it was not necessary to store a mass identifier for each species, nor was it necessary to carry the mass in the equations to be evaluated. The mass was normalized to unity and it could simply be dropped from the calculation. The introduction of multiple species has complicated the algorithm in several ways. It is of course necessary to identify the specie type of each particle, which adds to storage requirements. There is also an increased amount of computation required in collision mechanics as there are more free parameters, such as mass and number of atoms per particle, that must be taken into account. The cost of including the effects of vibration are minimal due both to the efficiency of the previously mentioned algorithm and the relatively infrequent occurrence of vibrationally relaxing collisions.

Previous implementations of the algorithm were written in the C programming language with time critical parts duplicated in Cray-2 machine code. This had restricted efficient use of the code to the Cray-2. In the past year, however, C compilers for the Cray have evolved to the point where the time critical constructs in the algorithm can now be recognized and vectorized by the compiler with acceptable performance loss in comparison to hand generated machine code. Compiler generated code for elements of this algorithm appear to be no more than 30% slower than the increasingly complex hand generated machine code. Further development on assembler versions were abandoned for this reason, yielding a large gain in portability and simplifying continued development. Current performance estimates will be discussed in a later section

Geometry

Inherent in the current implementation of the method is the representation of geometries in a cubic Cartesian mesh. This was motivated by the idea that the resulting savings in CPU time and memory could be used on simulating larger numbers of particles and a finer overall mesh. The rejection of body-fitted co-

ordinates introduces complications in the representation of geometries. This problem was addressed by the creation of an algorithm to represent a body as a collection of planar facets associated with each mesh cell[10]. In the previous calculation the AFE was represented by the aeroshell forebody and was terminated by a flat plate on the afterbody. The flight vehicle will, however, have a hexagonally shaped afterbody attached to the flat plate that is designed to hold instrumentation, maneuvering thrusters, and a booster that will drive the vehicle into the atmosphere. After the booster burns out it will be ejected leaving the hexagonal afterbody with minor appendages. This afterbody without the appendages has been added to the current geometry as shown in figure 1. The serrated appearance of the body results from the representation of the surface as a planar facet for each mesh cell. In cells through which a sharp edge of the body passes, there can be at least two defining planes associated with the intersecting surfaces. In order to prevent arbitrary rounding of these edges one of the original body surfaces is chosen to define the body for the cell. This surface will inevitably extend beyond the original body contour by up to one cell dimension. This represents the level of resolution of the defining geometry. Particles that interact with the surface within a given cell will see the plane associated with that cell as depicted in figure 1. In this particular case, the serrations are in a region of expanding flow where the local mean free path approaches the size of a mesh cell and are expected, for this reason, to have no noticeable effect on the flow field.

Calculations

The flight regime to be simulated was chosen to correspond to the calculation of the previous paper. The altitude was nominally $100km$ where the mean free path is $10cm$ and the free stream temperature is $190K$. The Mach number was chosen to be 35.42. The aeroshell diameter is $4.25m$ which is represented by 45 mesh cells in the calculation. To maintain the specified Knudsen number of 0.0235 this dictates a mean free path in simulation units of 1.03 mesh cells. As mentioned in the description of the enhancements to the method, chemical reactions are not yet modeled, however multiple species are represented. The gas mixture was composed of 18.1% O_2, 78.3% N_2 and 3.6% O, corresponding approximately to the equilibrium composition of air at this altitude. The intermolecular potentials of all species were modeled with a ninth order power law which is expected to approximate the known viscous behavior. The critical temperature for the vibrational modes was set to $3390K$ for nitrogen and to $2270K$ for oxygen. The surface boundary conditions impose diffuse reflection and complete thermal accommodation to the fixed wall temperature of $1500K$. Relaxation of internal energy modes was controlled by fixed collision numbers of 5 for rotation and 50 for vibration.

The calculation is started with a relatively small number of particles in order to establish the gross flow field features. After the number of particles in the domain comes to approximately steady state, they are "cloned" and a number of time steps are run. This process is repeated until the desired particle density is reached or until the computing environment is no longer able to provide the code with additional memory. Statistical averages are then collected. The simulation discussed here contained about $4.25 \cdot 10^6$ particles that were averaged over alternate time steps, in order to improve the statistical independence of the sampling. The averaging was done over 600 steps for a total of 300 time samples. The free stream number density in the simulation was 8 particles/cell. Figure 2 shows a comparison of the normalized temperatures along the geometric stagnation streamline for both calculations. Because the original calculation was close to being in thermal equilibrium, the thermal energy content is represented here by a single temperature, namely that of translation. The static temperature ratio across a shock in a perfect gas with $\gamma = 1.4$ at this Mach number is 245. The peak value reached in the previous calculation was 238 which then decreased toward the body surface because of the constant temperature boundary conditions. For the current simulation the translational temperature ratio for the mixture normalized on the free stream temperature reaches a peak value of 260 showing an expected overshoot. The rotational and vibrational temperatures lag behind the translational temperature with negligible vibrational energy content at the stagnation point. Figure 3 shows local species concentrations normalized by their values in the free stream. Any deviation from one on this plot indicates mass diffusion of species because of varying molecular velocities corresponding to their differing molecular weights. Atomic oxygen, as the lightest and smallest specie, is depleted in the region of high translational temperature and high density as it dif-

fuses the easiest. Conversely, molecular oxygen, as the heaviest specie, shows a slight increase in concentration. Because there is no chemistry, the molecular weight of the mixture varies only slightly from the free stream value such that the stagnation point environment for the current calculation is similar to that in the previous calculation. The shock standoff distance at the stagnation point is similar, although the shock thickness is larger due to the softer intermolecular potential. This can be seen in figures 4a and 4b which show density contours in the symmetry plane of the vehicle for both calculations. Near the lower skirt of the body, before the flow has turned over the shoulder, the gas has been vibrationally excited. Careful comparison of the two figures shows the shock standoff in the current calculation is be smaller than in the "perfect gas" case as is to be expected. The equilibrium temperature field for the previous calculation and the translational temperature field for the current calculation are shown in the symmetry plane in figures 5a,b. Both are qualitatively similar and should be compared to figures 6 and 7 which show respectively the rotational and vibrational temperature contours for the mixture. The rotational temperature lags behind the translational temperature along streamlines in the flow path from the stagnation point down to the skirt of the body and begins to relax again in the wake. This reflects the rotational collision number of 5 used in these calculations. The vibrational temperature lags considerably behind both the translational and rotational temperatures due to the collision number of 50. The vibrational modes become significantly populated by the time the flow has reached the skirt. The flow then expands around the shoulder so quickly, however, that the lowered collision rate combined with the vibrational collision number is not sufficient to depopulate the vibrational modes which then become frozen into the wake. There is at this time no mechanism in the algorithm other than collisions to depopulate these states. Figures 8a,b show velocity vectors in the wake region plotted in the symmetry plane for both calculations. Although the flight parameters are the same for both calculations, only the previous simulation shows evidence of vortical structures in the wake. The most likely explanation for this difference is the softer intermolecular potential in the present calculation, which increases the viscosity of the gas and effectively lowers the Reynolds number of the flow. Some of the difference may, however, be due to the addition of the hexagonal afterbody.

Performance Estimates and Code Capabilities

The current multiple specie, three dimensional code is named PSim. In its present form it is capable of moving an average particle through a complete time step, including overhead (start-up and restore costs), in about 2.0μ seconds. This compares favorably to the single specie version which typically averaged about 1.7μ seconds on similar runs. As the code is still under development, some further improvement is expected. These times of course vary with collision rate but do not however vary with other problem complexities such as number of simulated species. In the case of the AFE simulation, the elevated collision rate results in times as large as 5μ seconds/particle/timestep.

The PSim code is complemented by a general plotting system called Cplot that was developed recently to support the very large data sets associated with particle simulations. In its current incarnation, Cplot performs data management and general expression evaluation on a remote supercomputer and provides graphical presentation and a very powerful user interface on any Silicon Graphics Iris 4D workstation. Communication is via the Unix sockets facility. The plot file has a very flexible format that will allow the system to continue to be useful as PSim continues to evolve.

Ongoing Developments

The are two primary classifications of developments involving the Stanford particle simulation method. First are those that deal with extending the physical modeling capabilities of the method and second are those aspects related to algorithm development and high performance implementation on various machine architectures.

Haas[14] has recently developed models for implementing the effects of chemical non-equilibrium and there is currently an effort to incorporate those models into the PSim code. There is work in progress concurrently that will allow the PSim code to run across a wider range of machine platforms. Specifically, extensions will allow the method to operate not only on vector oriented machines and single CPU mini/micro systems, but also on parallel machines such as the Intel iPSC-2/RX. It will also permit the PSim code to utilize more than one CPU on the currently targeted Cray systems.

References

1. "Direct Simulation Scheme Derived from the Boltzmann Equation. in. Monocomponent Gases", Nanbu, K., Journal of the Physical Society of Japan, Vol. 49, No. 5, November 1980.
2. "Review of Monte Carlo Methods in Kinetic Theory", Derzko, N.A., UTIAS Rev. 35, University of Toronto, 78pp, 1972.
3. "Application of the Monte Carlo Method to Heat Transfer in a Rarefied Gas" Haviland, J.K. and Lavin, M.L., Physics of Fluids, Vol. 5, pp.1399-1405, 1965.
4. "Direct Simulation of Gas Flows at the Molecular Level", Bird, G. A., Communications in Applied Numerical Methods, Vol. 4, pp.165-172, 1988.
5. "Applicability of the Direct Simulation Monte Carlo Method in Body-Fitted Coordinate System", Shimada, T., Abe, T., Proceedings of the Sixteenth International Symposium on Rarefied Gas Dynamics, July 10-16, 1988, Pasadena, California.
6. "Three Dimensional Flow Simulation about the AFE Vehicle in the Transitional Regime", Celenligil, M. C., Moss, J. N., and Blanchard, R. C., AIAA-89-0245, AIAA 27th Aerospace Sciences Meeting, January 9-12, 1989, Reno Nevada.
7. "Vectorization of a Particle Simulation Method for Hypersonic Rarefied Flow", McDonald, J. D. and Baganoff, D., AIAA-88-2735, AIAA Thermophysics, Plasmadynamics and Lasers Conference, June 27-29, 1988, San Antonio, Texas.
8. "A collision-selection rule for an efficient particle-simulation method for use on vector computers", Baganoff, D., and McDonald, J. D., Physics of Fluids A,. Vol. 2, Issue 7, pp.1248-1259, July 1990.
9. "Application of a Vectorized Particle Simulation Method in High-Speed Near-Continuum Flow", Woronowicz, M. and McDonald, J., D., AIAA-89-1665, AIAA 24th Thermophysics Conference, June 12-14, 1989, Buffalo, New York.
10. "Three Dimensional Discrete Particle Simulation of an AOTV", William J. Feiereisen, Jeffrey D. McDonald, AIAA- 89-1711, AIAA 24th Thermophysics Conference, Buffalo, New York, June 12-14 1989.
11. "A Computationally Efficient Particle Simulation Method suited to Vector Computer Architectures" , McDonald, J.D., Report SUDAAR 589, Department of Aeronautics and Astronautics, Stanford University, December 1989.
12. "Statistical Collision Model for Monte-Carlo Simulations of Polyatomic Gas Mixture", Borgnakke, C., and Larsen, P. S., Journal of Computational Physics, Vol. 18, p405, 1975.
13. "Monte-Carlo Simulations in an Engineering Context", Bird, G.A., in *Rarefied Gas Dynamics*, ed. Fisher, S. S., Vol. 74, Progress in Aeronautics and Astronautics, Part I, AIAA, New York, 1981, pp 239-255.
14. "Fundamentals of Chemistry Modeling Applicable to a Vectorized Particle Simulation", Haas, B., AIAA-90-1749, AIAA 25th Thermophysics Conference, Seattle, Washington, June 18-20,1990.

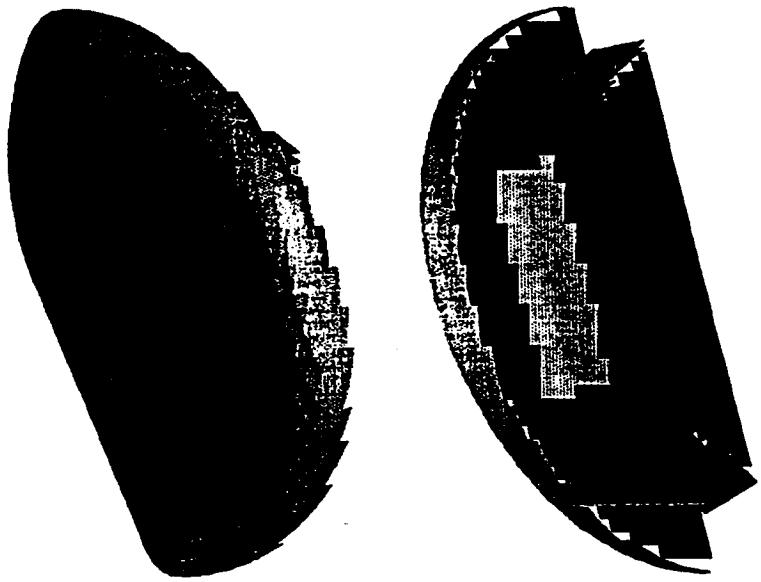

Figure 1. Two views of the AFE geometry as represented in the simulation.

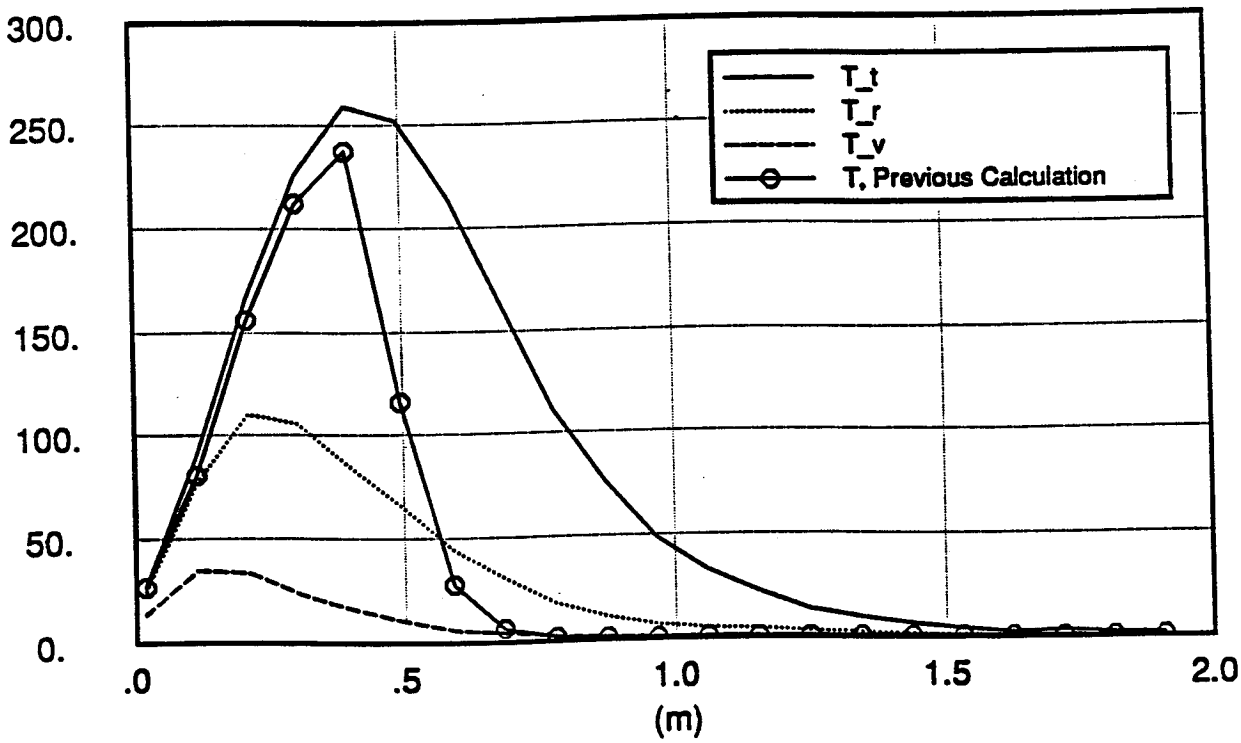

Figure 2. Temperatures along the stagnation streamline from the previous and current calculations, normalized on the free stream temperature of 190K.

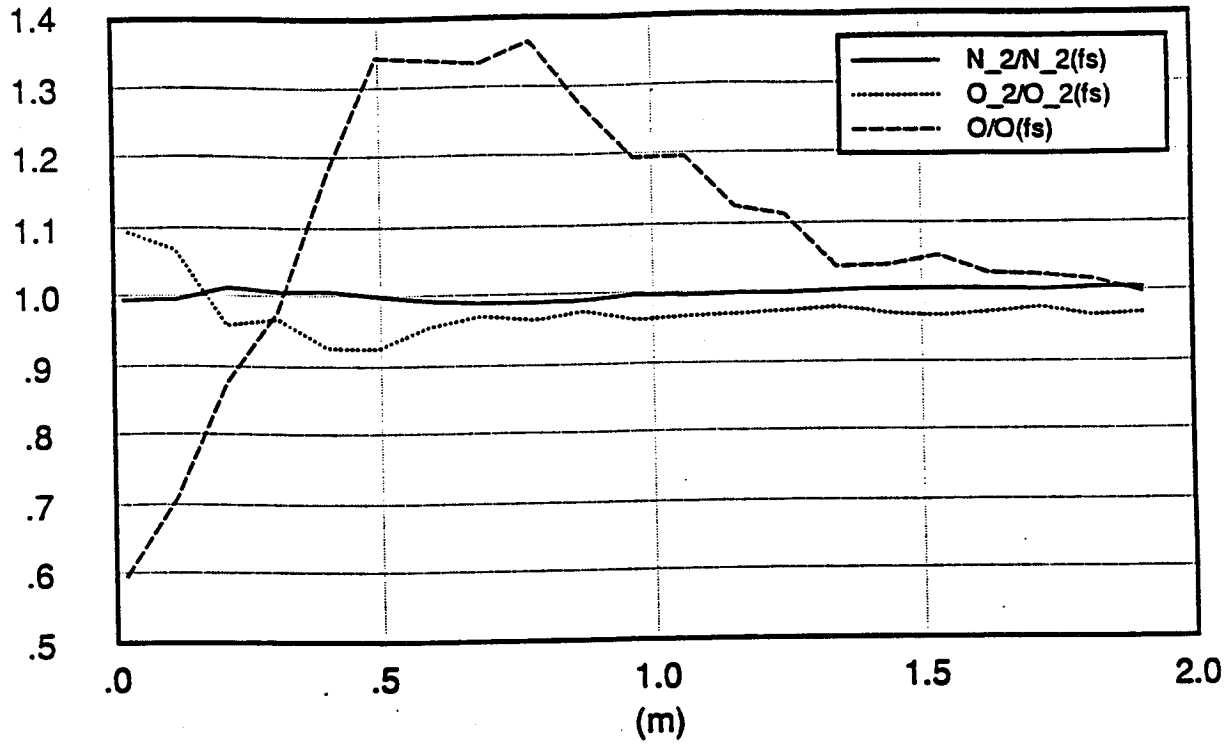

Figure 3. Species Concentrations along the stagnation streamline normalized on their values in the free stream.

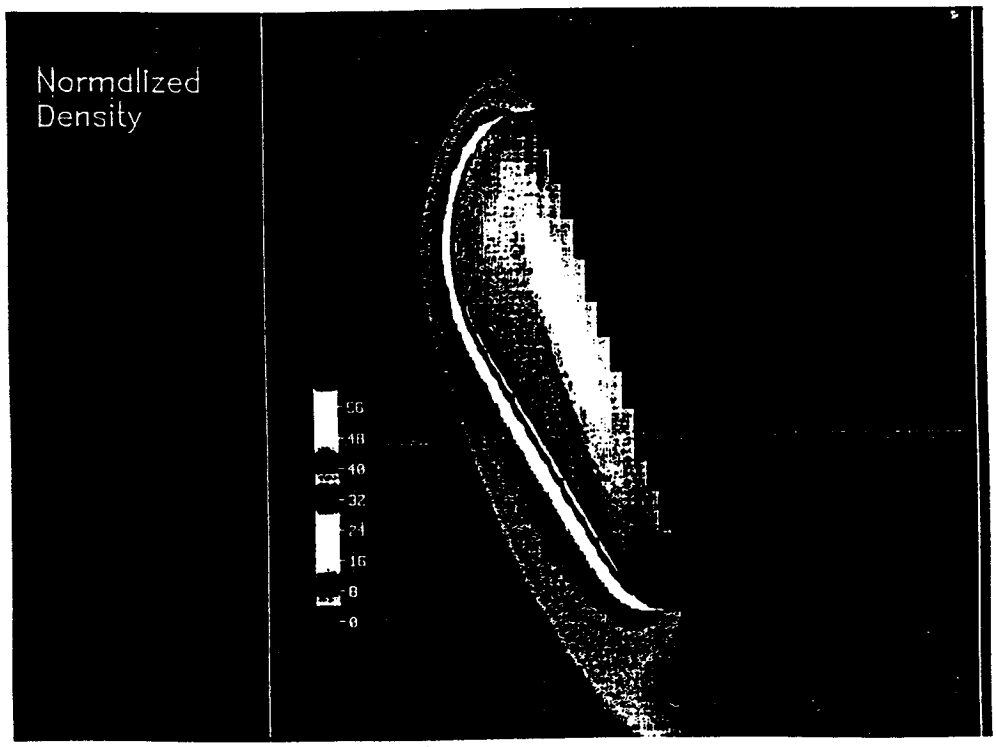

Figure 4a. Density contours in the symmetry plane for the previous calculation normalized on the free stream value.

Figure 4b. Density contours in the symmetry plane for the current calculation normalized on the free stream value.

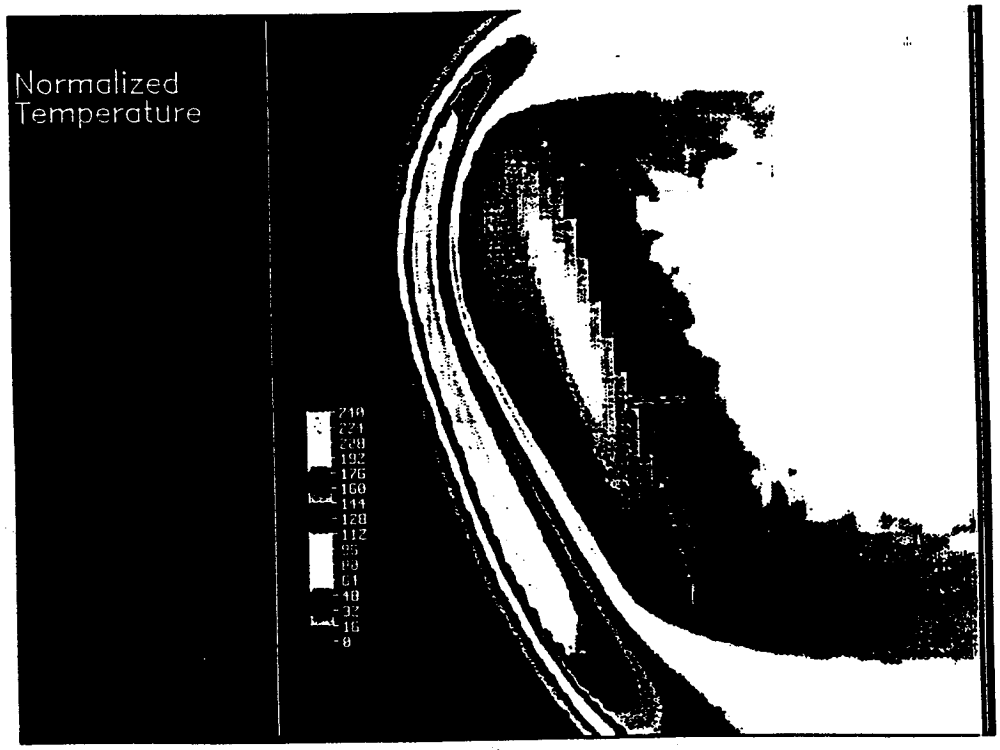

Figure 5a. Temperature contours in the symmetry plane for the previous calculation normalized on the free stream value.

Figure 5b. Translational temperature contours in the symmetry plane for the current calculation normalized on the free stream value.

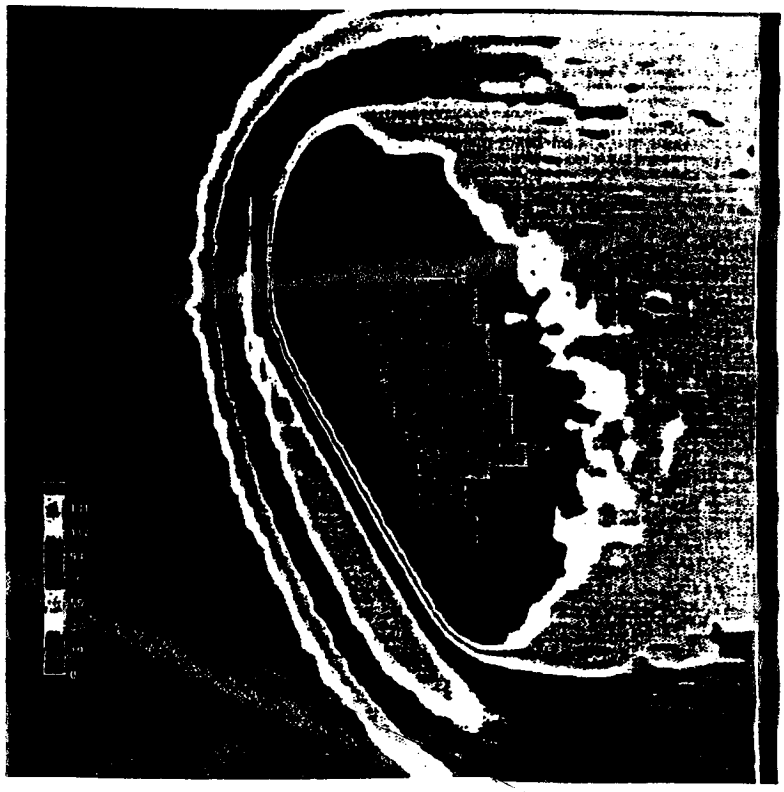

Figure 6. Rotational temperature contours in the symmetry plane for the current calculation normalized on the free stream value.

Figure 7. Vibrational temperature contours in the symmetry plane for the current calculation normalized on the free stream value.

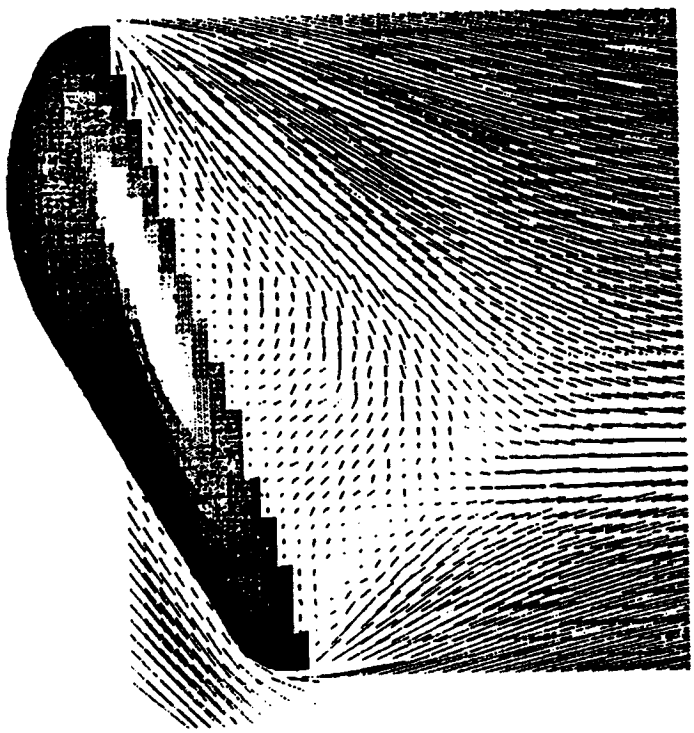

Figure 8a. Velocity vectors in the symmetry plane for the previous calculation.

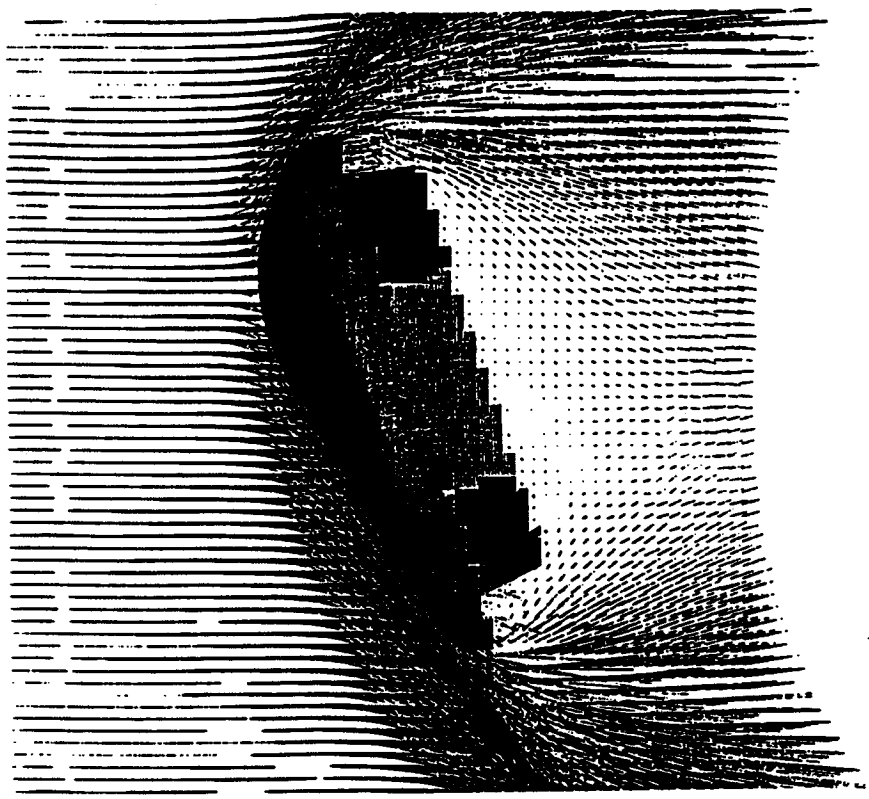

Figure 8b. Velocity vectors in the symmetry plane for the current calculation.